eBook Gratuit

APPRENEZ vagrant

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#vagrant

Table des matières

À propos 1

Chapitre 1: Commencer avec vagabond 2

Remarques 2

Liens utiles 2

Versions 2

Examples 3

Installation pour Windows avec prise en charge de VirtualBox et SSH 3

Installer VirtualBox 3

Installez Cygwin 3

Ajouter Cygwin à PATH 4

Installer Vagrant lui-même 4

Essai 4

Prochaines étapes 5

Projet LAMP 5

Télécharger une image de la boîte vaginale vers Amazon AWS AMI 11

La manière la plus simple d'avoir un Linux virtuel en quelques minutes (en 3 étapes) 11

Étape 1. 11

Étape 2. 12

Étape 3. 12

synchroniser tous les dossiers 12

Synchroniser les dossiers mais exclure certains dossiers 12

Chapitre 2: Des instantanés 14

Examples 14

Prenez un instantané 14

Script Bash pour supprimer tous les instantanés 14

Restaurer un instantané 14

Liste des instantanés 14

Restaurer un instantané sans approvisionner la boîte 14

Supprimer un instantané 14

Chapitre 3: Disposition 15
Syntaxe 15
Paramètres 15
Remarques 15
Examples 15
Configuration minimale 15
Lancer et provisionner la boîte 15
Lancer la boîte sans provisioning 16
Provisionner une boite de course 16
Shell Provider 16
Exécuter un script shell à partir d'un fichier (sans utilisation de l'inlining) 16
Chapitre 4: Fournisseurs 17
Examples 17
Définir le fournisseur par défaut avec la variable d'environnement 17
Définir le fournisseur par défaut dans Vagrantfile 17
Boîte de lancement dans Hyper-V 17
Boîte de lancement dans Docker 17
Zone de lancement dans VMWare Fusion 17
Zone de lancement dans VMWare Workstation 17
Boîte de lancement dans VirtualBox 17
Chapitre 5: Remise en service de la machine virtuelle en cours d'exécution 18
Examples 18
Commander 18
Chapitre 6: SSH 19
Examples 19
SSH à la boîte 19
SSH directement dans la boîte 19
Crédits 20

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: vagrant

It is an unofficial and free vagrant ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official vagrant.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Commencer avec vagabond

Remarques

Vagant un outil logiciel utilisé pour créer et configurer des environnements de développement virtuel. Cela fonctionne comme une enveloppe par rapport aux autres logiciels de virtualisation tels que VirtualBox ou VMware. Vagrant lui-même est un utilitaire de ligne de commande.

Les boîtes Vagrant sont configurées à l'aide de la configuration spéciale `Vagrantfile` écrite en Ruby, mais peuvent inclure des scripts de configuration supplémentaires écrits en bash, Chef ou Puppet. Les développeurs d'une équipe peuvent télécharger les fichiers de configuration Vagrant à partir d'une source commune et recréer le même environnement de développement localement.

Liens utiles

- Liste des boîtes vagabondes
- PuPHPet : constructeur puissant des boîtes vagabondes
- Packer : constructeur puissant des boîtes vagabondes

Versions

Version	Télécharger	Changelog	Date de sortie
1.8.5	Télécharger	Changelog	2016-07-18
1.8.4	Télécharger	Changelog	2016-06-13
1.8.3	Télécharger	Changelog	2016-06-10
1.8.1	Télécharger	Changelog	2015-12-24
1.8.0	Télécharger	Changelog	2015-12-21
1.7.4	Télécharger	Changelog	2015-07-17
1.7.3	Télécharger	Changelog	2015-07-10
1.7.2	Télécharger	Changelog	2015-01-06
1.7.1	Télécharger	Changelog	2014-12-12
1.7.0	Télécharger	Changelog	2014-12-10
1.6.5	Télécharger	Changelog	2014-09-04
1.6.4	Télécharger	Changelog	2014-09-02

Version	Télécharger	Changelog	Date de sortie
1.6.3	Télécharger	Changelog	2014-05-29
1.6.2	Télécharger	Changelog	2014-05-12
1.6.1	Télécharger	Changelog	2014-05-08
1.6.0	Télécharger	Changelog	2014-05-06

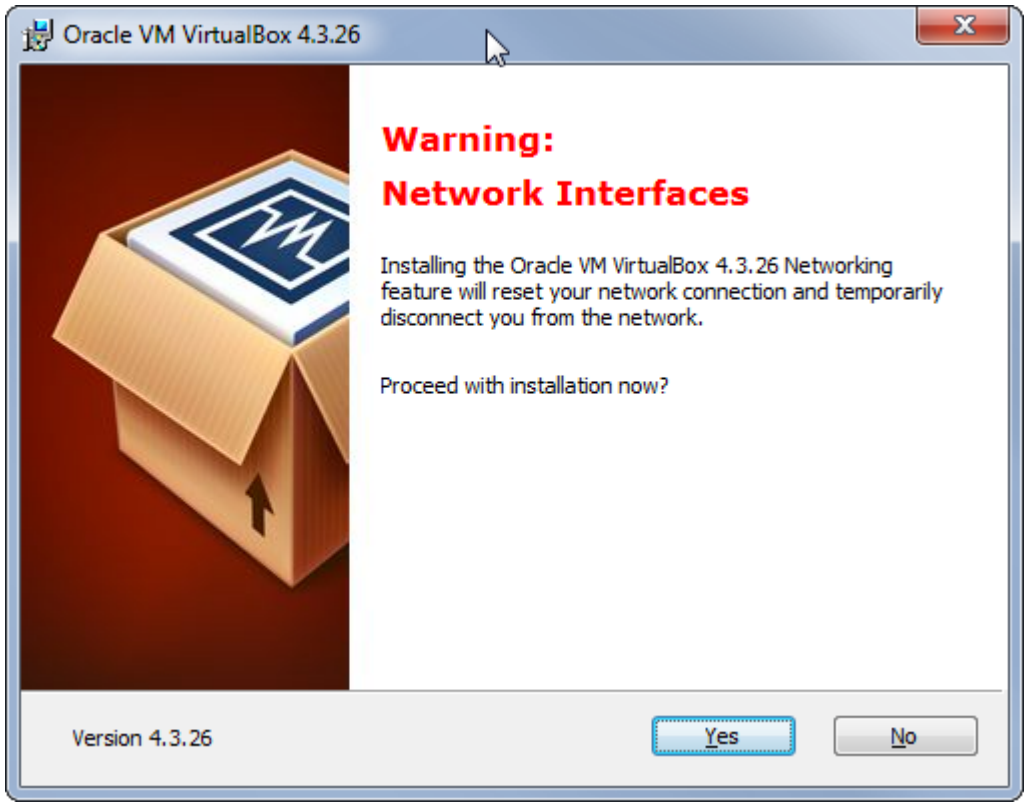
Examples

Installation pour Windows avec prise en charge de VirtualBox et SSH

Pour utiliser Vagrant sur la plate-forme Windows, vous devez d'abord installer un logiciel de virtualisation et un outil de ligne de commande ssh. Cet exemple utilisera freeware VirtualBox et Cygwin.

Installer VirtualBox

Téléchargez la dernière version de VirtualBox à partir de la page de téléchargement officielle et exécutez le fichier téléchargé. Notez que pendant l'installation, vous perdrez temporairement la connexion réseau.

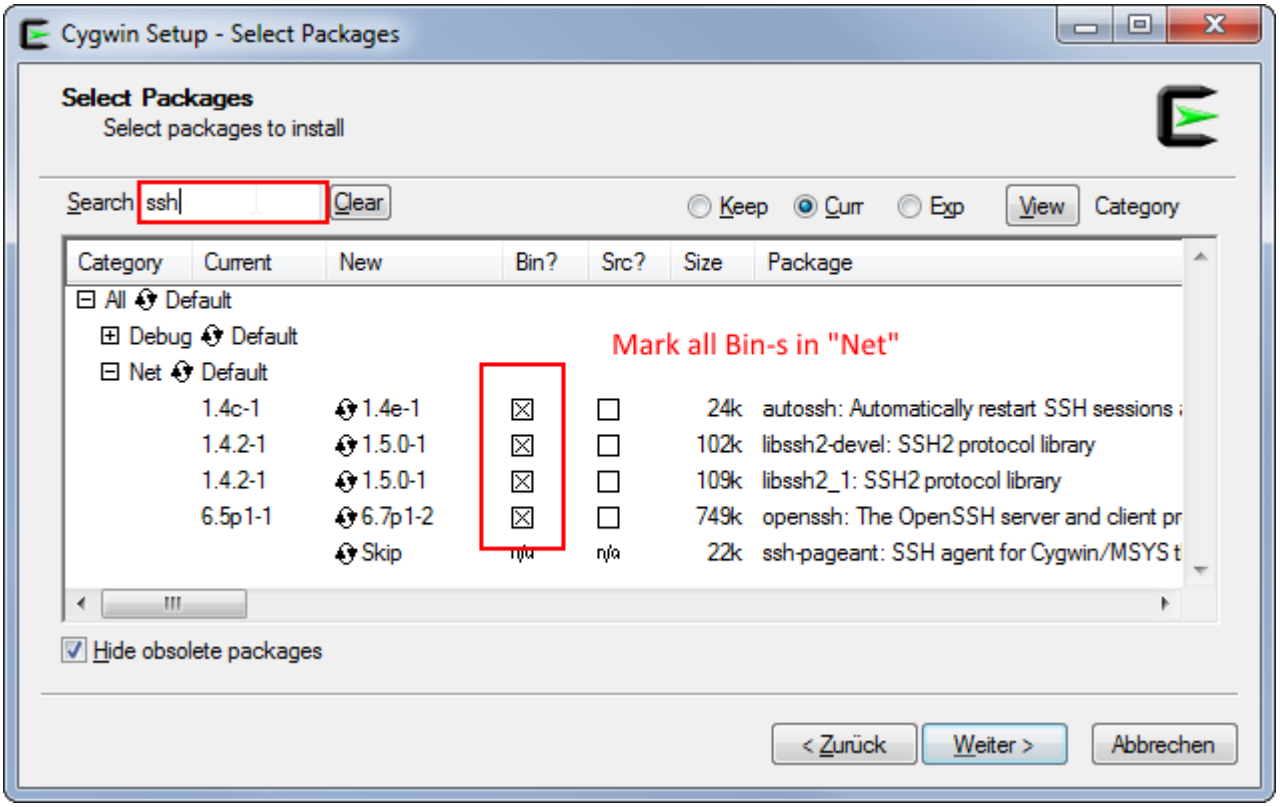


Vous devez également laisser Oracle installer des pilotes supplémentaires.

Installez Cygwin

Obtenez-le de cygwin.com et lancez l'installation jusqu'à ce que vous obteniez la page "Sélectionner les paquets".

Nous avons seulement besoin de ssh bin-s d'ici:



Ajouter Cygwin à PATH

Vous devez également ajouter le dossier `C:\cygwin64\bin` à la variable d'environnement Windows PATH. Pour que vous puissiez appeler la commande `ssh` depuis n'importe où.

Installer Vagrant lui-même

Téléchargez Vagrant à partir de vagrantup.com et suivez le guide d'installation pour l'installer. Vous devez redémarrer votre ordinateur après cela.

Essai

La technologie de virtualisation VTx / VTd doit être activée. (Vous pouvez le trouver dans le BIOS dans les options de sécurité)

Pour tester si tout installé correctement crée un dossier vide quelque part sur le disque dur, ouvrez la ligne de commande et accédez au dossier que vous venez de créer:

```
cd c:/path/to/your/folder
```

puis entrez

```
vagrant init hashicorp/precise64
vagrant up
```

Cela créera et lancera la VirtualBox exécutant Ubuntu 12.04 LTS 64 bits. Pour lancer la version 32 bits, utilisez `vagrant init hashicorp/precise32` . Si vous avez besoin d'une autre boîte, vous pouvez en trouver plus sur le site Web de hashicorp .

De plus, la commande `vagrant init` créera un fichier de configuration `Vagrantfile` dans le dossier actuel. Maintenant, vous pouvez simplement l'envoyer à quelqu'un d'autre et, lorsque cette personne appelle `vagrant up` la même machine virtuelle sera créée sur son PC.

Pour tester ssh après l'exécution réussie de ces deux commandes, exécutez cette commande dans le même dossier:

```
vagrant ssh
```

Si tout s'est bien passé, vous vous retrouverez dans la machine virtuelle connectée en tant qu'utilisateur `vagrant` .

Prochaines étapes

Vous pouvez arrêter la VM avec

```
vagrant halt
```

ou le supprimer avec

```
vagrant destroy
```

Vous trouverez plus de boîtes avec les instructions pour les installer sur la page vagrantbox.es .

Projet LAMP

Dans cet exemple, un environnement de développement de projet LAMP personnalisé est créé avec Vagrant.

Tout d'abord, vous devrez installer Virtual Box et Vagrant .

Ensuite, créez un dossier `vagrant` dans votre répertoire personnel, ouvrez votre terminal et remplacez le répertoire actuel par le nouveau répertoire `vagrant` . Maintenant, exécutez `vagrant init` . Un fichier `Vagrantfile` sera créé à l'intérieur et ressemblera à ceci:

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

# All Vagrant configuration is done below. The "2" in Vagrant.configure
# configures the configuration version (we support older styles for
# backwards compatibility). Please don't change it unless you know what
# you're doing.
Vagrant.configure(2) do |config|
  # The most common configuration options are documented and commented below.
  # For a complete reference, please see the online documentation at
  # https://docs.vagrantup.com.

  # Every Vagrant development environment requires a box. You can search for
  # boxes at https://atlas.hashicorp.com/search.
  config.vm.box = "base"

  # Disable automatic box update checking. If you disable this, then
  # boxes will only be checked for updates when the user runs
  # `vagrant box outdated`. This is not recommended.
  # config.vm.box_check_update = false

  # Create a forwarded port mapping which allows access to a specific port
  # within the machine from a port on the host machine. In the example below,
  # accessing "localhost:8080" will access port 80 on the guest machine.
  # config.vm.network "forwarded_port", guest: 80, host: 8080

  # Create a private network, which allows host-only access to the machine
  # using a specific IP.
  # config.vm.network "private_network", ip: "192.168.33.10"

  # Create a public network, which generally matched to bridged network.
  # Bridged networks make the machine appear as another physical device on
  # your network.
  # config.vm.network "public_network"

  # Share an additional folder to the guest VM. The first argument is
  # the path on the host to the actual folder. The second argument is
  # the path on the guest to mount the folder. And the optional third
  # argument is a set of non-required options.
  # config.vm.synced_folder "../data", "/vagrant_data"

  # Provider-specific configuration so you can fine-tune various
  # backing providers for Vagrant. These expose provider-specific options.
  # Example for VirtualBox:
  #
  # config.vm.provider "virtualbox" do |vb|
  #   # Display the VirtualBox GUI when booting the machine
  #   vb.gui = true
  #
  #   # Customize the amount of memory on the VM:
  #   vb.memory = "1024"
  # end
  #
  # View the documentation for the provider you are using for more
  # information on available options.

  # Define a Vagrant Push strategy for pushing to Atlas. Other push strategies
  # such as FTP and Heroku are also available. See the documentation at
  # https://docs.vagrantup.com/v2/push/atlas.html for more information.
  # config.push.define "atlas" do |push|
  #   push.app = "YOUR_ATLAS_USERNAME/YOUR_APPLICATION_NAME"
```

```
  # end

  # Enable provisioning with a shell script. Additional provisioners such as
  # Puppet, Chef, Ansible, Salt, and Docker are also available. Please see the
  # documentation for more information about their specific syntax and use.
  # config.vm.provision "shell", inline: <<-SHELL
  #   sudo apt-get update
  #   sudo apt-get install -y apache2
  # SHELL
end
```

Ajoutez ou décommentez les lignes suivantes en modifiant le fichier ci-dessus:

```
config.vm.box = "hashicorp/precise64"
config.vm.network :forwarded_port, guest: 80, host: 8080
config.vm.network :private_network, ip: "192.168.33.10"
config.ssh.forward_agent = true
if Vagrant::Util::Platform.windows?
  config.vm.synced_folder "./", "/vagrant"
else
  config.vm.synced_folder "./", "/vagrant", type: "nfs"
end
# https://stefanwrobel.com/how-to-make-vagrant-performance-not-suck
config.vm.provider "virtualbox" do |v|
  host = RbConfig::CONFIG['host_os']

  # Give VM 1/4 system memory & access to all cpu cores on the host
  if host =~ /darwin/
    cpus = `sysctl -n hw.ncpu`.to_i
    # sysctl returns Bytes and we need to convert to MB
    mem = `sysctl -n hw.memsize`.to_i / 1024 / 1024 / 4
  elsif host =~ /linux/
    cpus = `nproc`.to_i
    # meminfo shows KB and we need to convert to MB
    mem = `grep 'MemTotal' /proc/meminfo | sed -e 's/MemTotal://' -e 's/ kB//'`.to_i / 1024 /
4
  else # sorry Windows folks, I can't help you
    cpus = 2
    mem = 1024
  end

  v.customize ["modifyvm", :id, "--memory", mem]
  v.customize ["modifyvm", :id, "--cpus", cpus]
end
```

Modifiez votre fichier `hosts` pour rediriger votre domaine souhaité vers la VM vagabond. Pour Linux, il s'agit de `/etc/hosts` , pour Windows `C:\Windows\System32\Drivers\etc\hosts` ; et ajoutez cette ligne:

```
192.168.33.10 vagrantServer.com
```

Bien sûr, vous pouvez remplacer `vagrantServer.com` par n'importe quel nom.

Maintenant, il est temps de créer le fichier `bootstrap.sh` (dans le répertoire `vagrant`). Ce script sera exécuté chaque fois que la machine virtuelle est générée à partir de zéro. Lisez attentivement les commentaires:

```
#!/usr/bin/env bash

# .ssh/authorized_keys (you will need to create a `.ssh` directory inside the `vagrant` one
and add a file named `authorized_keys` with the public keys of the users who have access to
the repository and may use this environment).
# You also will have to grant access to those public keys from the Github account, Bitbucket,
or whatever you're using.
cat /vagrant/config/authorized_keys >> /home/vagrant/.ssh/authorized_keys
if ! [ -d /root/.ssh ]; then
  mkdir /root/.ssh
fi
cp /vagrant/config/authorized_keys /root/.ssh
chmod 600 /root/.ssh/authorized_keys

# Install packages
apt-get update
apt-get install -y python-software-properties
add-apt-repository ppa:ondrej/php5 -y
apt-get update
apt-get install -y curl nano apache2 php5 php5-mysql php5-curl php5-gd php5-intl php5-mcrypt
git

# Apache2 run with user vagrant
APACHEUSR=`grep -c 'APACHE_RUN_USER=www-data' /etc/apache2/envvars`
APACHEGRP=`grep -c 'APACHE_RUN_GROUP=www-data' /etc/apache2/envvars`
if [ APACHEUSR ]; then
    sed -i 's/APACHE_RUN_USER=www-data/APACHE_RUN_USER=vagrant/' /etc/apache2/envvars
fi
if [ APACHEGRP ]; then
    sed -i 's/APACHE_RUN_GROUP=www-data/APACHE_RUN_GROUP=vagrant/' /etc/apache2/envvars
fi
sudo chown -R vagrant:www-data /var/lock/apache2

# Set user/password to mysql previously to installation
# Replace rootMysqlPassword with your desired MySQL root password
debconf-set-selections <<< 'mysql-server mysql-server/root_password password
rootMysqlPassword'
debconf-set-selections <<< 'mysql-server mysql-server/root_password_again password
rootMysqlPassword'

# Install mysql
apt-get update
apt-get install -y mysql-server mysql-client

# Link /vagrant (sync_folder) to apache directory (/var/www)
if ! [ -L /var/www ]; then
  rm -rf /var/www
  ln -fs /vagrant /var/www
fi

# Install composer
curl -sS https://getcomposer.org/installer | php
mv composer.phar /usr/local/bin/composer

# Composer example. Uncoment to istall phpunit
#composer global require "phpunit/phpunit=3.7.*" --prefer-source

# Create mysql database (replace "vagrantDB" with any desired database name, and
"rootMysqlPassword" with the password set above)
mysql -u root -prootMysqlPassword -v -e "CREATE USER 'developer'@'%' IDENTIFIED BY 'dev';
CREATE SCHEMA vagrantDB;
```

```
GRANT ALL ON vagrantDB TO 'developer'@'%';"

# Uncomment to set default database fixtures based on `/vagrant/config/vagrantDBFixtures.sql`
file.
#mysql -u root -prootMysqlPassword -v vagrantDB < /vagrant/config/vagrantDBFixtures.sql

##################################################
################ THIS IS OPTIONAL ################
##################################################

# Install nodejs
curl -sL https://deb.nodesource.com/setup | sudo bash -
apt-get install -y nodejs

# Install npm packages
npm install -g npm
npm install -g bower
npm install -g forever
npm install -g gulp

# Set accepted license before install java
echo debconf shared/accepted-oracle-license-v1-1 select true | sudo debconf-set-selections
echo debconf shared/accepted-oracle-license-v1-1 seen true | sudo debconf-set-selections

# Install java7
apt-get install -y oracle-java7-installer oracle-java7-set-default

##################################################
################## END OPTIONAL ##################
##################################################

# Generate ssh key without passphrase
ssh-keygen -f /root/.ssh/id_rsa -t rsa -N ""

# Add bitbucket and github to known hosts
touch /root/.ssh/known_hosts
ssh-keyscan -H bitbucket.org >> /root/.ssh/known_hosts
ssh-keyscan -H github.com >> /root/.ssh/known_hosts

# Source: https://gist.github.com/winhamwr/7122947
# Sleep until we can successfully SSH into Bitbucket.
# Uses doublinng backoff while waiting
# with_backoff() adapted from http://stackoverflow.com/a/8351489
# Retries a command a configurable number of times with backoff.
#
# The retry count is given by ATTEMPTS (default 5), the initial backoff
# timeout is given by TIMEOUT in seconds (default 1.)
#
# Successive backoffs double the timeout.
#generatedKey="`cat /root/.ssh/id_rsa.pub`"
echo -n "Generate a SSH key (https://help.github.com/articles/generating-ssh-keys/)
and add it to your Bitbucket account (Profile -> SHH keys) to continue. "

with_backoff() {
  local max_attempts=${ATTEMPTS-5}
  local timeout=${TIMEOUT-1}
  local attempt=0
  local exitCode=0

  while [ $attempt -lt $max_attempts ]
  do
```

```
    set +e
    "$@"
    exitCode=$?
    set -e

    if [ $exitCode -eq 0 ]
    then
      break
    fi

    echo "Failure! Retrying in $timeout.." 1>&2
    sleep $timeout
    attempt=$(( attempt + 1 ))
    timeout=$(( timeout * 2 ))
  done

  if [ $exitCode -ne 0 ]
  then
    echo "You've failed me for the last time! ($@)" 1>&2
  fi

  return $exitCode
}

ATTEMPTS=${ATTEMPTS:-5}

export ATTEMPTS
with_backoff ssh -T git@bitbucket.org;

# Clone repositories (replace "yourProjectName" and "yourProjectRepository" with your project
data)
cd /var/www
rm -rf yourProjectName/
git clone yourProjectRepository

# Add server names to /etc/hosts (replace "vagrantServer.com" with the domain set above)
echo -e '\n127.0.0.1       vagrantServer.com' >> /etc/hosts

# Enable apache modes
a2enmod rewrite

# Copy sites-available file (you need to add the Apache configuration file for the desired
domain in `config/sites-available`. Replace "vagrantServer.conf" with the desired name)
cp /vagrant/config/sites-available/vagrantServer.conf /etc/apache2/sites-available/

# Remove html from document root
sed -i 's/\/var\/www\/html/\/var\/www/g' /etc/apache2/sites-available/*
service apache2 restart

# Enable sites (replace "vagrantServer.conf" with the above file name)
a2ensite vagrantServer.conf

# Install ruby, compass and sass (Optional)
apt-get install -y rubygems
gem install compass
npm install -g sass

# Pull the repo
cd /var/www/yourProjectName
git pull --all
```

Une fois coché et enregistré le fichier ci-dessus, rendez-vous à nouveau à votre terminal, changez le répertoire en cours pour le `vagrant` vous avez créé avant et tapez `vagrant up` . La machine virtuelle sera créée et le fichier boostrap exécuté à partir de la machine virtuelle afin que tous les éléments nécessaires soient copiés / installés. Une fois terminé, vous pouvez ouvrir votre navigateur et aller sur `vagrantServer.com` (ou le nom que vous lui avez donné) et vous devriez voir le contenu de la VM vagabond.

Vous pourrez également éditer vos fichiers de projet via le répertoire `vagrant/yourProjectName` et tous les fichiers `vagrant/yourProjectName` le répertoire `vagrant` seront partagés et synchronisés entre votre hôte et la machine virtuelle vagabonde.

Télécharger une image de la boîte vaginale vers Amazon AWS AMI

Vous avez une boîte de dialogue locale que vous souhaitez télécharger sur Amazon AWS. Tout d'abord, vous devez créer un fichier `.box` :

```
vagrant package --base my-virtual-machine
```

Cette étape devrait prendre un certain temps en fonction de la taille de votre image. Ensuite, vous devez obtenir l'image `.vmdk` partir du fichier `.box` :

```
gunzip -S .box package.box
tar xf package
```

Après cette étape, vous devriez avoir 4 nouveaux fichiers: `package` , `box-disk1.vmdk` , `Vagrantfile` et `box.ovf` . Maintenant, pour télécharger sur AWS. En supposant que vous ayez un compte AWS , créez un compartiment S3 pour stocker l'image sur les serveurs d'Amazon. Vous allez avoir besoin de l'EC2 CLI d'Amazon pour la prochaine étape (car vous ne pouvez pas le faire via la console pour autant que je sache):

```
ec2-import-instance box-disk1_1.vmdk -f VMDK -t t2.micro -a x86_64 -b <S3-bucket-name> -o
$AWS_ACCESS_KEY -w $AWS_SECRET_KEY -p Linux
```

Ce résultat de cette commande devrait prendre un certain temps - il télécharge le gros fichier image vers S3, mais la commande elle-même retourne plus rapidement. Vous pouvez vérifier la progression de l'importation à l'aide de la commande `ec2-describe-conversion-tasks` .

Une fois que cela est terminé, vous verrez une instance de votre boîte s'exécuter dans la console AWS. Cependant, vous ne pourrez peut-être pas y accéder car il ne possède pas d'adresse IP publique et / ou ne possède pas de fichier `.pem` associé. L'étape suivante consiste donc à créer une AMI à partir de l'instance. Pour créer une AMI, arrêtez l'instance (ne terminez pas!) Et cliquez avec le bouton droit sur l'instance et accédez à `Image` -> `Create Image` . Cela devrait prendre un certain temps aussi. Vous pouvez vérifier la progression dans la vue AMI de la console. Une fois qu'il a terminé, lancez une instance en utilisant l'AMI en y attachant un fichier de clé `.pem` , puis vous pouvez y `ssh` et vous êtes prêt.

La manière la plus simple d'avoir un Linux virtuel en quelques minutes (en 3 étapes)

Étape 1.

Dans votre machine hôte (Windows / Linux / OSX), créez un `my_project` vide `my_project` .

Étape 2.

Créez un fichier nommé `Vagrantfile` avec ceci:

```
Vagrant.configure("2") do |config|
  config.vm.box = "gbarbieru/xenial" #An Ubuntu 16.04 based image
  config.vm.hostname = "my_project"
  config.vm.network :private_network, ip: "172.16.123.10"
end
```

Étape 3.

Exécutez votre machine:

```
host$ vagrant up
host$ vagrant ssh
virtual$ cd /vagrant
```

Terminé!

Remarques:

- Vous souhaitez peut-être attribuer une autre adresse IP.
- Si votre hôte est Windows, vous voudrez peut-être `ssh` de `putty` . Vous pouvez le faire en vous connectant au nom d'hôte `127.0.0.1` et au port `2222` . Le nom d'utilisateur est `vagrant` et le mot de passe est `vagrant` .

synchroniser tous les dossiers

Pour synchroniser tous les dossiers dans les deux sens, insérez-le dans votre `Vagrantfile`

```
config.vm.synced_folder "my-project1", "/home/vagrant/my-project1"
```

Synchroniser les dossiers mais exclure certains dossiers

Pour synchroniser tous les dossiers dans les deux sens, insérez-le dans votre fichier Vagrant:

```
config.vm.synced_folder "my-project1", "/home/vagrant/my-project1", type: "rsync",
 :rsync__exclude => ['my-project1/mini_project2/target,my-project1/mini_project2/target,my-
project1/mini_project3/target']
```

Tous les dossiers cibles sont exclus de la synchronisation.
La synchronisation ne se produit que sur les `vagrant up` et sur les `vagrant reload` .
Pour synchroniser chaque modification de votre hôte avec l'invité, vous devez utiliser:

```
vagrant rsync-auto
```

Lire Commencer avec vagabond en ligne: https://riptutorial.com/fr/vagrant/topic/1026/commencer-avec-vagabond

Chapitre 2: Des instantanés

Examples

Prenez un instantané

```
vagrant snapshot save mysnapshot
```

Script Bash pour supprimer tous les instantanés

```
#!/bin/bash

NO_SNAPSHOTS="No snapshots have been taken yet"
SNAPSHOT_OUTPUT=$(vagrant snapshot list | grep "${NO_SNAPSHOTS}")

if [ -z "${SNAPSHOT_OUTPUT}" ]; then
    echo "Found some snapshots, going to remove them"
    for SNAPSHOT in $(vagrant snapshot list); do
        vagrant snapshot delete "${SNAPSHOT}"
    done
else
    echo "No snapshots found"
fi
```

Restaurer un instantané

```
vagrant snapshot restore mysnapshot
```

Liste des instantanés

```
vagrant snapshot list
```

Restaurer un instantané sans approvisionner la boîte

```
vagrant snapshot restore --no-provision mysnapshot
```

Supprimer un instantané

```
vagrant snapshot delete mysnapshot
```

Lire Des instantanés en ligne: https://riptutorial.com/fr/vagrant/topic/2483/des-instantanes

Chapitre 3: Disposition

Syntaxe

- config.vm.provision "shell", inline: COMMANDS
- config.vm.provision "shell", chemin: "relativePath / script.sh"

Paramètres

Paramètre	Détails
COMMANDES	Les commandes shell à exécuter. Peut être une chaîne (par exemple `"echo \"Hello, World!\""`) Ou une variable de chaîne (par exemple `$setup`).

Remarques

Le provisionnement est utilisé pour configurer automatiquement une machine virtuelle. Il est effectué automatiquement lorsqu'une machine virtuelle est créée pour la première fois (en utilisant `vagrant up`). Il peut également être ré-exécuté plus tard en utilisant la `vagrant provision` .

Examples

Configuration minimale

Vagrantfile

```
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/trusty64"
  config.vm.provision "ansible" do |ansible|
    ansible.playbook = "vagrant-playbook.yml"
  end
end
```

vagrant-playbook.yml

```
---
- hosts: default
  tasks:
    - name: Say hello
      debug:
        msg: 'Hello, World'
```

Lancer et provisionner la boîte

```
vagrant up
```

Par défaut, la boîte sera provisionnée.

Lancer la boîte sans provisioning

```
vagrant up --no-provision
```

Provisionner une boite de course

```
vagrant provision
```

Shell Provider

Le shell provisioner exécute un script shell lors du provisionnement.

```
$setup = <<SETUP
# You can write your shell script between here ...
sudo echo "Hello, World!" > /etc/motd.tail
# ... and here.
SETUP

Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/trusty64"
  config.vm.provision "shell", inline: $setup
end
```

Exécuter un script shell à partir d'un fichier (sans utilisation de l'inlining)

```
# provision/bootstrap-controller.sh : path and shell filename from vagrantfile location
config.vm.define "configcontroller" do |controller|
...
controller.vm.provision :shell do |shell|
    shell.path = "provision/bootstrap-controller.sh"
end
...
```

Lire Disposition en ligne: https://riptutorial.com/fr/vagrant/topic/3091/disposition

Chapitre 4: Fournisseurs

Examples

Définir le fournisseur par défaut avec la variable d'environnement

Introduire la variable d'environnement `VAGRANT_DEFAULT_PROVIDER`

```
export VAGRANT_DEFAULT_PROVIDER=vmware_fusion
```

Définir le fournisseur par défaut dans Vagrantfile

```
Vagrant.configure("2") do |config|
  # ... other config up here

  config.vm.provider "vmware_fusion"
end
```

Boîte de lancement dans Hyper-V

```
vagrant up --provider hyperv
```

Boîte de lancement dans Docker

```
vagrant up --provider docker
```

Zone de lancement dans VMWare Fusion

```
vagrant up --provider vmware_fusion
```

Zone de lancement dans VMWare Workstation

```
vagrant up --provider vmware_workstation
```

Boîte de lancement dans VirtualBox

```
vagrant up --provider virtualbox
```

VirtualBox est le fournisseur par défaut pour une configuration Vagrant vanille.

Lire Fournisseurs en ligne: https://riptutorial.com/fr/vagrant/topic/2482/fournisseurs

Chapitre 5: Remise en service de la machine virtuelle en cours d'exécution

Examples

Commander

```
vagrant reload --provision
```

Lire Remise en service de la machine virtuelle en cours d'exécution en ligne: https://riptutorial.com/fr/vagrant/topic/6056/remise-en-service-de-la-machine-virtuelle-en-cours-d-execution

Chapitre 6: SSH

Examples

SSH à la boîte

```
vagrant ssh
```

SSH directement dans la boîte

```
vagrant ssh-config >> ~/.ssh/config
ssh default
```

Lire SSH en ligne: https://riptutorial.com/fr/vagrant/topic/6609/ssh

Crédits

S. No	Chapitres	Contributeurs
1	Commencer avec vagabond	Community, Daniel Käfer, dragon788, ha_1694, Joerg, Manolo, mrtuovinen, Robin, Ruslan Bes, theoretisch, Xavi Montero
2	Des instantanés	mrtuovinen
3	Disposition	dragon788, mrtuovinen, Robin, snonov
4	Fournisseurs	Ates Goral, mrtuovinen
5	Remise en service de la machine virtuelle en cours d'exécution	Lernkurve
6	SSH	Daniel Käfer, mrtuovinen